PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-097204

(43)Date of publication of application : 08.04.1997

(51)Int.Cl. G06F 12/00
G06F 12/00
G06F 9/06
G06F 9/44
G06F 13/00
G06F 15/16

(21)Application number : 08-077336

(22)Date of filing : 29.03.1996

(71)Applicant : SUN MICROSYST INC

(72)Inventor : HAPNER MARK W
SNYDER ALAN

(30)Priority

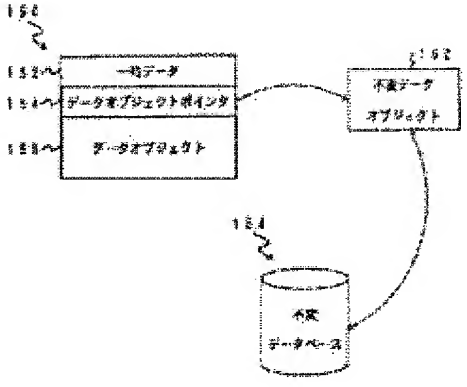
Priority number : 95 414770 Priority date : 31.03.1995 Priority country : US

(54) METHOD AND DEVICE FOR PROVIDING TRANSPARENT PERSISTENCE IN DISTRIBUTED OBJECT OPERATION ENVIRONMENT

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a method and device for providing a transparent persistence in a distributed object operation environment.

SOLUTION: A persistent data storage mechanism having a persistent data object 162 which supplies a servant object and in which persistent data is stored and a persistent data object pointer 154 which indirectly shows the persistent data object is provided. The value of a data object 158 is judged, the data object pointer 154 is arranged in the servant object, the data object pointer 154 is substituted for the persistent data pointer, thereby distributed object having transparent persistence data support is generated.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平9－97204

(43)公開日　平成９年(1997)４月８日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ			技術表示箇所
Ｇ０６Ｆ	12/00	５４７		Ｇ０６Ｆ	12/00	５４７Ａ	
		５４５				５４５Ａ	
	9/06	５３０			9/06	５３０Ｔ	C11-18
	9/44	５３５			9/44	５３５	
	13/00	３５７			13/00	３５７Ｚ	

審査請求　未請求　請求項の数19　ＯＬ　（全 16 頁）　最終頁に続く

(21)出願番号　特願平8－77336

(22)出願日　平成８年(1996)３月29日

(31)優先権主張番号　０８／４１４７７０

(32)優先日　1995年３月31日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　595034134
サン・マイクロシステムズ・インコーポレイテッド
Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ，　Ｉｎｃ．
アメリカ合衆国カリフォルニア州94043－1100・マウンテンビュー・ガルシアアベニュー　2550

(72)発明者　マーク　ダブリュー．　ハプナー
アメリカ合衆国，　カリフォルニア州　95125，　サン　ノゼ，　ブルックス　アベニュー　595

(74)代理人　弁理士　長谷川　芳樹　（外４名）

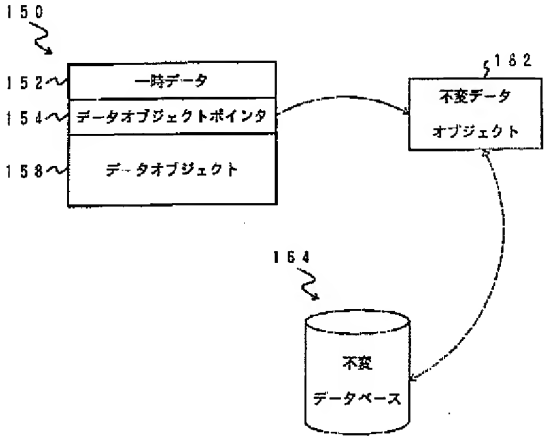
最終頁に続く

(54)【発明の名称】　分散オブジェクト操作環境において透明性のある不変性を提供する方法および装置

(57)【要約】

【課題】　分散オブジェクト操作環境において、透明性のある不変性を提供する、方法および装置を提供する。

【解決手段】　サーバントオブジェクトを供給し、不変データが格納される不変データオブジェクトと、前記不変データオブジェクトを間接的に示す不変データオブジェクトポインタとを有する不変データ格納機構を供給して、データオブジェクトの値を判定し、データオブジェクトポインタをサーバントオブジェクト内に配置し、データオブジェクトポインタを不変データポインタで置き換えて、透明性のある不変データサポートを有する分散オブジェクトを作成する。



【特許請求の範囲】

【請求項１】　分散オブジェクトシステムで使用される、透明性のある不変データサポートを有する分散オブジェクトを提供する、コンピュータに実装される方法であって、

ａ）一時データを格納可能な関連データオブジェクトと、前記関連データオブジェクトを間接的に示すデータオブジェクトポインタとを有し、データオブジェクトクラスからデータオブジェクトを受け継ぐサーバントオブジェクトを提供するステップと、

ｂ）不変データが格納される不変データオブジェクトと、前記不変データオブジェクトを間接的に示す不変データオブジェクトポインタとを有する不変データ格納機構を提供するステップと、

ｃ）前記データオブジェクトの値を判定するステップと、

ｄ）前記データオブジェクトポインタを前記サーバントオブジェクト内に配置し、前記データオブジェクトポインタを前記不変データポインタで置き換えて、透明性のある不変データサポートを有する分散オブジェクトを作成するステップと、

備えるコンピュータに実装される方法。

【請求項２】　前記不変データ格納機構は、前記不変データオブジェクトと結び付いた不変データベースを備える、請求項１記載のコンピュータに実装される方法。

【請求項３】　前記不変データ格納機構は、不変データマネージャを備える、請求項２記載のコンピュータに実装される方法。

【請求項４】　前記判定するステップは、前記サーバントオブジェクト上での機能を拡張し、前記データオブジェクトクラス内の前記データオブジェクトポインタを判定するステップを備える、請求項１記載のコンピュータに実装される方法。

【請求項５】　前記配置するステップは、前記データオブジェクトポインタと同一の値を含むアドレス空間の位置を得るために、前記サーバントオブジェクトのメモリ位置を探査するステップを備える、請求項４記載のコンピュータに実装される方法。

【請求項６】　前記配置するステップは、前記データオブジェクトの前記アドレスの唯一性を保証するために、前記サーバントオブジェクトの前記メモリ空間の全体を探査するステップを更に有する、請求項５記載のコンピュータに実装される方法。

【請求項７】　オブジェクトリクエストブローカとオブジェクトアダプタ機構とを使用して分散オブジェクトが通信する分散オブジェクトとクライアントが配置されたメモリを有する、複数のネットワーク接続されたコンピュータを含む分散オブジェクトシステム上のコンピュータのメモリ内に、前記分散オブジェクトが配置される、請求項１記載のコンピュータに実装される方法。

【請求項８】　前記分散オブジェクトシステム上のクライアントによる、前記サーバントオブジェクトの発動に応答して、請求項１のａ）ないしｄ）のステップが実行される、請求項７記載のコンピュータに実装される方法。

【請求項９】　前記分散オブジェクト上での前記サーバントオブジェクトの作成および据付けの間に、請求項１のａ）ないしｄ）のステップが実行される、請求項７記載のコンピュータに実装される方法。

【請求項１０】　請求項１のコンピュータに実装される方法によって形成された、透明性のある不変データサポートを有する分散オブジェクト。

【請求項１１】　透明性のある不変データサポートを有する分散オブジェクト作成するコンピュータシステムであって、

ａ）データオブジェクトクラス中へデータスキーマをコンパイルするデータコンパイラと、

ｂ）コンパイルされた実装中へ実装ファイルをコンパイルする実装コンパイラと、

ｃ）一時データスペース、データオブジェクトポインタ、およびデータオブジェクトを含み、前記データオブジェクトクラスから前記データオブジェクトポインタを継承するサーバントオブジェクトを作成可能なサーバントオブジェクトコンストラクタを作成する最終段階コンパイラと、

ｄ）不変データベースと前記不変データベースのイメージである不変データオブジェクトとを含む不変データストレージ機構と、

ｅ）前記データオブジェクトと前記データオブジェクトのアドレススペースとの値を見出すロケータ、および、前記アドレススペースの値を前記不変データオブジェクトを示すポインタ値とスワップする置換機構を含む透明性のある不変性機構とを備えるコンピュータシステム。

【請求項１２】　前記不変データストレージ機構は、前記不変データオブジェクトポインタを提供する不変ストレージマネージャを備える、請求項１１記載のコンピュータシステム。

【請求項１３】　前記一時データスペース、前記データオブジェクトポインタ、および前記データオブジェクトは、コンピュータ内の隣接領域を占める、請求項１１記載のコンピュータシステム。

【請求項１４】　前記ロケータは、前記データオブジェクトポインタの値を判定する拡張機構と、前記サーバントオブジェクトのメモリスペース内で前記データオブジェクトポインタの前記メモリアドレスを探す探査機構とを備える、請求項１３記載のコンピュータシステム。

【請求項１５】　オブジェクトリクエストブローカ機構およびオブジェクトアダプタ機構を使用して、通信を行う分散オブジェクトと、クライアントが置かれたメモリを有する、ネットワーク接続された複数のコンピュータ

を備える分散オブジェクトシステム上のいずれかのコンピュータのメモリ内に、前記分散オブジェクトが置かれる、請求項１１記載のコンピュータシステム。

【請求項１６】　前記透明性のある不変性機構は、前記サーバの発動に応答して活性状態とされる、請求項１５記載のコンピュータシステム。

【請求項１７】　前記透明性のある不変性機構は、前記サーバの作成に応答して活性状態とされる、請求項１５記載のコンピュータシステム。

【請求項１８】　請求項１２記載のコンピュータシステムによって形成された、透明性のある不変データサポートを有する分散オブジェクト。

【請求項１９】　分散オブジェクトシステム上での使用に好適な分散オブジェクトであって、
一時データを格納する一時データスペースと、
データオブジェクトと、
データオブジェクトからの読み込み位置と、不変データベースのイメージである不変データオブジェクトへの書き込み位置とを間接的に示す不変データポインタと、
備える分散オブジェクト。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、分散コンピューティングシステム、クライアントーサーバコンピューティングおよびオブジェクト指向プログラミングの分野に係わり、特に、分散オブジェクトシステムにおいて透明性のある不変性（transparent persistence）を提供する方法および装置を含む。

【０００２】

【従来の技術および発明が解決しようとする課題】ここ数年来、従来のプログラミング手法で開発されたソフトウェアの納期遅れと予算超過の傾向が強まるにつれて、オブジェクト指向プログラミング手法への関心が高まっている。従来のプログラミング手法に伴う１つの問題は、多くの課題に関して、設計及び保守がしばしば著しく困難な手順モデルと「線形」コードとに重点が置かれていることに起因している。一般的に、従来の手法を用いて作成された大きなプログラムは「脆い（brittle）」。すなわち、わずかな変更でさえも、プログラミングコードの全要素が影響を受ける。従って、ユーザの要望に応じたソフトウェアの小規模な変更でも、全プログラムの大幅な再設計と書き換えとが必要になることがある。

【０００３】オブジェクト指向プログラミング戦略は、こうした問題の回避に向いている。その理由は、オブジェクト手法は、手順よりもむしろデータの取り扱いに重点を置くので、プログラマに現実世界の問題をモデル化するにあたって、より直感的なアプローチを提供するからである。更に、オブジェクトは関連データと手順とをカプセル化しており、オブジェクトのインターフェイスを介してのみ関連するデータおよび手順へのアクセスを可能にすることによって、これらの情報がプログラムの他の部分から隠されている。従って、オブジェクトのデータおよび／または手順への変更は、プログラムの他の部分から比較的隔離された状態にある。こうして、特定のオブジェクトコードの変更が、他のオブジェクトのコードに影響を及ぼすことがないので、従来の方法を用いて書かれたコードに比べて保守の容易なコードが提供される。更に、オブジェクトは本来モジュール的な性質であり、個々のオブジェクトを種々のプログラムで再使用することができる。したがって、プログラマは「試行正（tried and true）」のオブジェクトのライブラリを作成し、個々のアプリケーションにオブジェクトを反復して使用することができる。このことにより、ソフトウェアの信頼性が高まると共に、信頼性のあるプログラミングコードが繰り返し用いられるので、開発時間が短縮される。

【０００４】オブジェクト指向方法論の分野における最近の進歩は、コンピュータネットワークを介して相互に接続された複数のコンピュータにまたがる分散オブジェクトオペレーティング環境の実現の中で見られた。ここで使用する意味での用語「分散オブジェクト」または「オブジェクト」は、インターフェイスを介する動作によって操作できるコードとデータのカプセル化されたパッケージをいう。したがって、分散オブジェクトは、オブジェクト指向プログラミング（ＯＯＰ）の分野の当業者には、従来通りのプログラミングオブジェクトを定義する基本的特性を含むものと見られるであろう。しかし、分散オブジェクトは、２つの重要な特徴を有することによって従来のプログラミングオブジェクトとは異なる。第１に、分散オブジェクトは多言語的（multilingual）である。すなわち、分散オブジェクトのインターフェイスは、多様な異なるプログラミング言語にマップ（map）できるインターフェイス定義言語（ＩＤＬ）を使用して定義される。そのようなインターフェイス定義言語の１つがオブジェクトマネージメントグループ（Object Manegement Group）のＩＤＬである。第２に、分散オブジェクトは場所独立性がある。すなわち、分散オブジェクトは、ネットワーク内のどの場所にも配置され得るのである。このことは、通常は単一のアドレス空間の中に存在する従来のプログラミングオブジェクトとは著しく対照的である。

【０００５】分散オブジェクト操作環境について更に詳しく述べると、分散オブジェクトは、他のオブジェクトへリクエストを送っているか、あるいはクライアントからのリクエストに回答しているかによって、オブジェクトクライアントあるいはオブジェクトサーバーとなり得る。分散オブジェクト環境においては、リクエストと回答は、オブジェクトの場所と状態を認識しているオブジェクトリクエストブローカ（Object Request Broker）

を介して行われる。そうしたＯＲＢに適したひとつのアーキテクチャが、共通オブジェクトリクエストブローカアーキテクチャ（ＣＯＲＢＡ）仕様によって提供される。ＣＯＲＢＡ仕様は、サービスをリクエストするオブジェクトへサーバーオブジェクトがサービスを提供することのできる分散クライアントサーバー環境におけるオブジェクトに関して分散コンピューティング環境を定義するため、ＯＭＧ（Object Management Group）によって開発された。

【０００６】クライアントが目標オブジェクト（target object）をコールする場合、目標オブジェクトがリクエストされたサービスを実行できることを保証するため、特定の手順が実行される必要がある。こうした手順は、目標オブジェクトの識別および配置、（もし、必要ならば）目標オブジェクト側でのサーバーオブジェクトの起動、（もし、必要ならば）目標オブジェクトの活性化、そして最後に、目標オブジェクトとの接続を形成してコールを送る。ＯＲＢは、オブジェクトアダプタ（ＯＡ）とともに、クライアントと目標オブジェクトと共働して、これらのサービスを実行する。

【０００７】分散オブジェクトのライフサイクルの観点から、オブジェクトは、一時オブジェクトと不変オブジェクトという２つのカテゴリのいずれか一方となる。オブジェクトの属性が一時的か不変的かという議論を行う場合、オブジェクトの状態の属性が一時的か不変的かが参照される。オブジェクト指向プログラミング（ＯＯＰ；object oriented programming）の当業者にはなじみのように、オブジェクトは、実行可能なコードと状態という２の要素によって記述されている。実行可能なコードは、本質的に、それによってオブジェクトが動作する命令であり、オブジェクトの「振る舞い（behavior）」を定義する。状態は、単に、コードでは無いデータのような、オブジェクトの残余の部分である。

【０００８】大多数の先行技術では、一時オブジェクトに関して開示している。通常、一時データは短いライフサイクルを有し、単一のホストコンピュータプロセスとのみ結び付いて存在する。この結果、ホストコンピュータプロセスが中止した場合に、ホストコンピュータプロセス中に存在する全ての一時オブジェクトが中止する。したがって、あるプロセスから他のプロセスへの一時オブジェクトの同一性の連続性は存在しなくなる。一時オブジェクトが単一のプロセスのみ結び付いているので、一時オブジェクトは、本質的に、それらの位置を変化することができない。したがって、一時オブジェクトのアドレスは変化しないので、一時オブジェクトは、「インモービル（immobile）」オブジェクトとも呼ばれる。一時オブジェクトのみを含むオブジェクト種を使用するプログラマにとっては、インスタンスからインスタンスへのオブジェクト状態が保護されるにすぎない。

【０００９】一方、不変オブジェクトは単一のプロセスにのみ結び付くのでは無く、不変オブジェクトのアドレスとメモリ位置は時間が経過しても変更できる（たとえば、不変オブジェクトは多くの「ライフ（life）」サイクルを有する）。不変オブジェクトでは、あるプロセスから他のプロセスへの一時オブジェクトの同一性の連続性が存在する。要するに、不変オブジェクトは、状態がオブジェクトの特定のインスタンスの寿命よりも長命のオブジェクトである。不変オブジェクトが、オブジェクト開発者に対して多くの利益をもたらすことは、明らかである。不幸にも、従来の不変オブジェクトの実装に関する戦略は、上記のように、オブジェクト開発者に対して、十分に満足のいく解答を提供できていなかった。

【００１０】不変オブジェクトを提供する、最も未成熟な戦略では、おそらく、オブジェクト内のプログラマコードは、ハードディスクのようなパーマネントストレージ媒体からのデータの読み込みやデータの書き込み（すなわち、管理）である。この戦略は単純であるが、少なくとも２つの欠点がある。第１に、オブジェクト開発者は、夫々の不変オブジェクトの中へこのデータ管理を実装するにあたって、負荷を担わなければならない。第２に、この戦略では、不要なオーバヘッドを覚悟しなければならない。例を挙げると、夫々のオブジェクトは、不変データサポートについて事前に記述されたコードを含まなければならない。更に、データファイルの構造が非常に複雑なものとなり、データの読み出しや書き込みのために詳細な分析が必要となる。こうしたデータ管理の全ては、オブジェクトによる実行時間内に終了されねばならず、システム資源の利用の点で、高価なものになってしまう結果をもたらす。更に、分散オブジェクト操作環境に複雑性が付与され、オブジェクト開発者が不変性戦略を実装することが非常に困難となる。これは、部分的には、システムの外に配置された多くの代理オブジェクトを有する分散オブジェクトの属性によるものである。

【００１１】不変オブジェクトを提供する他のアプローチでは、不変性は、現存のオブジェクトのプログラミング言語中に導入される。１つのアプローチでは、現存のオブジェクトのプログラミング言語の拡張が、データベースの機能性へのインターフェイスを提供する。例えば、このアプローチを使用する拡張された言語に関するコンパイラは、データベースの使用に対して適合するコマンドの特別セットを許容しなければならない。リレーショナルデータベースは、オブジェクト状態が通常に見出される形態での格納データに関しての設計はなされていないので、ジレンマを引き起こす。オブジェクト指向データベースは利用可能であるが、オブジェクト指向データベースによって作成されたオブジェクトはＣＯＲＢＡオブジェクトではなく、ＣＯＲＢＡガイドラインの利益を享受できない。いずれにしても、先行技術の戦略では、データの不変性を注意深く管理しなければならない

ので、プラグラマには不当な負荷が課されることになる。

【００１２】図１を参照して、ＣＯＲＢＡ仕様に従って設計された特定のオブジェクトについて議論する。図１に、一時データ１２２、データオブジェクトポインタ１２４、およびデータオブジェクト１２０を含む、分散オブジェクトクラスのサーバントオブジェクト１２０を示す。背景として、クラスは、それからオブジェクトが作成され得るテンプレートである。オブジェクト指向プログラミングの分野の当業者にとっては理解されるように、クラスの新たなインスタンスが作成されると、インスタンス変数に関するメモリが割り当てられる。サーバントオブジェクト１２０は、分散オブジェクトの１つのインスタンスである。図１には、サーバントオブジェクト１２０のメモリ割り当てが示されている。このメモリ領域が連続領域であることに注意されたい。一時データ１２２は、オブジェクトが一時データを使用するためのみに必要な状態（例えば、変数）を含むサーバントオブジェクト１２０のメモリの一部分である。

【００１３】一時データは、サーバントオブジェクト１２０のインスタンスによってのみ使されるスクラッチパッド領域と類似している。データオブジェクト１２８は、オブジェクトが不変状態としての使用を意図するインスタスのメモリの一部である。まず、サーバントオブジェクト１２０が作成されると、オブジェクトコンストラクタが、当初は空の一時データおよびデータオブジェクトとともに、サーバントオブジェクト１２０を作成する。（作成中に実行される）初期化機能および（活動中に実行される）活性化機能は、要請された初期化を実行しなければならない。以前に議論したように、オブジェクト開発者は、データオブジェクト１２８を保守し、呼び起こすために、なんらかのタイプの不変性戦略と共働しなければならず、データは、作成および活性化／非活性化のサイクルを通して保守される。通常の方法では、タスクを実行する、複雑な初期化機能および活性化機能を書き込まねばならない。

【００１４】ＣＯＲＢＡ仕様は、不変データを管理するオブジェクト開発者を援助する１つの可能な手段を提供する。オブジェクトアダプタデータベース（object adapterdatabase；ＯＡＤ）内で、ＯＲＢは、各登録オブジェクトごとに、参照データと呼ばれる、最大１キロバイト（１Ｋバイト）の不変データ部を保守する。活性化において、オブジェクトは、ＯＡＤ内の適当な参照データの配置に関するキーとして使用できる参照キーを獲得する。開発者は、データオブジェクト１２８に関連する不変データを格納するために、１Ｋバイトの参照データを利用することができるが、１Ｋバイトという制限は、多くの応用にとっては不向きであろう。したがって、オブジェクト開発者は、より複雑な不変性戦略を実装しなければならない。例えば、不変データを含むメモリ空間をオブジェクトに示すポインタのリストは有効であろう。ＣＯＲＢＡ仕様に従うにも拘らず、オブジェクト開発者は、不変性戦略の実装にあたっては、負荷を負うことになる。

【００１５】従来のプログラミングオブジェクトに関する不変データストレージを実装する方法および装置と、不変性機構がオブジェクト開発者にとって透明性があるような分散オブジェクトが必要なのである。このためには、不変性機構が透明性のある不変性となるであろう。透明性のある不変オブジェクトは、プログラマが明示的な状態のロードやセーブを実装しなくともよいので、オブジェクト指向プログラマの作業が容易になるであろう。したがって、透明性のある不変オブジェクトサポートを提供する方法および装置が必要なのである。

【００１６】

【課題を解決するための手段】本発明は、分散オブジェクト操作環境におけるオブジェクトに対して、不変データサポートを提供する方法および装置を提供することによって、上記の制限を克服する。特に、以下に記載の方法および装置は、プログラマに対して相対的に透明性のある不変性に関するサポートを提供することによって、オブジェクト内での不変性の生成に関連する負荷からオブジェクトプログラマを解放する。

【００１７】本発明は、分散オブジェクトシステム上で使用に関して、透明性のある不変性データサポートを有する分散オブジェクトを提供するコンピュータに実装される方法を含む。本発明の方法は、一時データが格納され得る関連データオブジェクト（associated data object）と、関連データオブジェクトを間接的に指示するデータオブジェクトポインタ（data object pointer）とを有するサーバントオブジェクト提供する。サーバントオブジェクトは、データオブジェクトクラス（data object class）からデータオブジェクトを継承する。また、本発明の方法は、不変（persistent）データが格納される不変データオブジェクトと、不変データオブジェクトを間接的に指示する不変データオブジェクトポインタとを有する不変データ格納機構を提供する。更に、データオブジェクトポインタの値は判定され、データオブジェクトポインタは、サーバントオブジェクト内に置かれ、不変データポインタに置き換えて、透明性のある不変データサポートを有する分散オブジェクトを作成する。

【００１８】また、本発明は、透明性のある不変データサポートを有する分散オブジェクトを作成するコンピュータシステムを提供する。本発明のコンピュータシステムは、データオブジェクトクラス中へデータスキーマをコンパイルするデータコンパイラと、コンパイルされた実装（implementation）中へ実装ファイルをコンパイルする実装コンパイラとを備える。更に、本発明のコンピュータシステムは、一時データスペース、データオブジ

エクトポインタ、およびデータオブジェクトを含み、データオブジェクトクラスからデータオブジェクトポインタを継承するサーバントオブジェクトを作成可能なサーバントオブジェクトコンストラクタを作成する最終段階コンパイラを備える。更に、不変データベースと不変データベースのイメージである不変データオブジェクトとを含む不変データストレージ機構を備える。そして、データオブジェクトとデータオブジェクトのアドレススペースとの値を見出すロケータ、および、アドレススペースの値を不変データオブジェクトを示すポインタ値とスワップする置換機構を含む透明性のある不変性機構を備える。

【００１９】また、本発明は、分散オブジェクトシステム上での使用に好適な分散オブジェクトを含む。この分散オブジェクトは、一時データを格納する一時データスペースと、データオブジェクトと、データオブジェクトからの読み込み位置と、不変データベースのイメージである不変データオブジェクトへの書き込み位置とを間接的に示す不変データポインタと備える。

【００２０】

【発明の実施の形態】本発明は、オブジェクト指向プログラミング（ＯＯＰ）に基づく分散オペレーティング環境に係わり、特に、分散オペレーティング環境において、透明性のある不変性を提供する方法および装置に関するものである。以下、用語の定義に続いて、まず、本発明の装置を例示する。次に、本発明に好適な、コンピュータプロセス、ネットワーク、およびコンピュータシステムを例について議論し、引き続いて、本発明の装置とデータ構造のいくつかの実施例を詳細に説明することによって、更に、本発明の方法面の詳細な説明をして、本発明の方法と装置を詳細に説明する。

【００２１】Ｉ．用語の定義

ここで使用する意味での用語「分散オブジェクト」または「オブジェクト」はオブジェクトと関連する定義済インターフェイスを介して作用によって操作できるコードとデータのカプセル化されたパッケージをいう。したがって、分散オブジェクトは、当業者にとっては、従来通りのプログラミングオブジェクトを定義する基本的特性を含むものと見えるであろう。しかし、分散オブジェクトは、２つの重要な特徴を含むことによって、従来のプログラミングオブジェクトとは異なる。第１に、分散オブジェクトは多言語的（multilingual）であるという点である。分散オブジェクトのインターフェイスは、種々の異なるプログラミング言語にマッピングできるインターフェイス定義言語を使用して定義される。そのようなインターフェイス定義言語の１つがＯＭＧのＩＤＬである。第２に、分散オブジェクトは場所独立性がある。すなわち、分散オブジェクトはネットワーク内のどの場所にも配置できるのである。これは、通常はクライアントと同じアドレス空間に存在している従来のプログラミングオブジェクトとは著しく対照的である。分散オブジェクトは、他のオブジェクトへリクエストを送信しているのか、または他のオブジェクトからのリクエストに応答しているのかによって、オブジェクトクライアントまたはオブジェクトサーバの何れかとなる。リクエストおよび応答は、オブジェクトの場所と状態を認識しているオブジェクトリクエストブローカ（Object Request Broker；ＯＲＢ）を介して行われる。

【００２２】「分散オブジェクトシステム」または「分散オブジェクトオペレーティング環境」は、ＯＲＢを介して通信する複数の分散オブジェクトを含むシステムを表わす。

【００２３】「オブジェクトレファレンス」または「オブジェレフ（obfref）」は、別のオブジェクトに対するポインタを含むオブジェクトである。更に、オブジェレフはサブオブジェクトを識別するために使用できるメモリの一部（「サブオブジェクト識別子」）を含むことができる。サブオブジェクト識別子を除いて、オブジェクトレファレンスの作成と定義は、当業者には周知である。

【００２４】ここで定義する「クライアント」は、第２のクライアントへリクエストを送信する実体をいう。このモデルにおいては、第２のオブジェクトを「サーバオブジェクト」または「ターゲットオブジェクト」という。したがって、クライアントはサーバから作用または実装実行を発動する。分散オブジェクト環境では、オブジェクトが多言語性を有するという条件があるため、クライアントは実装実行プログラミング言語の知識を持つ必要がなく、実装実行もクライアントのプログラミング言語の知識を持つ必要がない。分散オブジェクト環境におけるクライアントとサーバは、インターフェイス定義言語によって通信するだけで良い。先に述べた通り、クライアントからサーバへのリクエストおよびクライアントへのサーバの応答は、ＯＲＢによって処理される。クライアントとサーバは同じプロセスにあっても良く、同じホストコンピュータにあっても良く、あるいは２つの異なるホストコンピュータにあっても良いことを指摘しておくべきである。

【００２５】「オブジェクトインターフェイス」は、オブジェクトが提供する作用、属性および例外の仕様である。分散オブジェクトの場合のオブジェクトインターフェイスは、ＩＤＬを使用して書き込まれるのが好ましい。前述のように、オブジェクトはそのインターフェイスを介してトランザクションを実行する。したがって、インターフェイスを使用すると、使用されるプログラミング言語を認識しているオブジェクトがトランザクション中のオブジェクトの方法およびデータを定義する必要がなくなる。

【００２６】情報のパケットを「マーシャル（整頓）する」とは、この情報を共有メモリまたはネットワーク通

信ラインを介して転送するように準備することである。これは、しばしば、使用するネットワーク通信プロトコルに従ってデータを編成することを意味する。

【００２７】情報のパケットを「アンマーシャル（整頓解除）する」とは、本質的にマーシャリング手順を逆にすることであって、非ネットワーク環境において意味をもつフォーマットにおいてデータを作成することである。

【００２８】「透明性のある不変性（transparent persistence）」とは、オブジェクト開発者に対して、不変データの格納と管理とを提供する機構のことである。

【００２９】ＩＩ．透明性のある不変性

本発明によれば、分散オブジェクト操作環境内のプログラミングオブジェクトに透明性のある不変性を提供する種々の方法および装置が提供される。ここで、「透明性のある不変性」とは、オブジェクト開発者に対して、不変データの格納と管理とを提供する機構のことである。いくつかの実施例において、オブジェクトのインスタンス変数の構造の透明性のある構成（特に、オブジェクトの限定された不変メモリを示すアドレス情報）と、オブジェクトの関連した不変データの自動的な管理とを説明する。いくつかの実施例においては、（分散オブジェクトに対して外部にあるとともに透明性を有する）不変ストレージマネージャが、不変オブジェクトデータを保守する。

【００３０】ひとつの実施例では、透明性のある不変性戦略は、以下のように、オブジェクトコンパイル（obuject compilation）の属性と継承の原理（the principle ofinheritance）を利用する。オブジェクト開発者が、データスキーマ（data schema）中のデータ変数を宣言すると、直接的な関係でそれが実行される。しかしながら、オブジェクト開発者のデータスキーマがデータオブジェクトクラス（dataobject class）中にコンパイルされると、直接的な関係は、コンパイラによって間接的なものに変換される。（「間接的な指示（indirection）」は、情報、ポインタなどの集合であり、オブジェクトにクライアントエンティティを示すことのできる、例えばロケータオブジェクトのようなソースにクライアントエンティティを示す。（クライアントが地理的な位置をリクエストすると、間接的な指示は、クライアントに現在の地図上の位置を示すか、多分、地理に詳しい個人の電話番号をクライアントに連絡するのと類似している。）

したがって、直接的な関係は変換され、オブジェクトの活動中の関係は、コンパイラによって挿入され、オブジェクト開発者が意識いないデータポインタとなるであろう。データポインタは、他のインスタンス変数に隣接するメモリ中のデータオブジェクトを間接的に示す。オブジェクトが活性化されると、オブジェクトクラスの単一のインスタンスが生成され、インスタンスは、データオブジェクトを示すデータポインタを含むオブジェクトクラスの振る舞いと属性を継承する。本発明では、オブジェクト活動中に不変データオブジェクトのレプリカを作成し、データオブジェクトに対する初期データポインタを不変データオブジェクトに対するポインタに置き換える。

【００３１】オブジェクトは、オブジェクトクライアントであってもよいし、オブジェクトサーバであってもよく、他のオブジェクトにリクエストをおくるか、クライアントからのリクエストに応答するかで、オブジェクトクライアントであるか、オブジェクトサーバであるかが決まる。分散オブジェクト環境においては、リクエストおよび応答は、オブジェクトの位置と状態とに通じているオブジェクトリクエストブローカ（ＯＲＢ）を介してなされる。こうしたＯＲＢの実装に好適な一つのアーキテクチャが、コモンオブジェクトリクエストブローカアーキテクチャ（Commom Object Request Broker Architecture）仕様によって供給される。ＣＯＲＢＡ仕様は、サービスをリクエストするクライアントへサーバーオブジェクトがサービスを提供することのできる分散クライアントサーバー環境におけるオブジェクトに関して分散コンピューティング環境を定義するため、ＯＭＧ（ObjectManagement Group）によって開発された。以下の説明では、「オブジェクト」と「分散オブジェクト」は互換性をもって仕様される。しかしながら、一般的には、分散オブジェクトは、分散オブジェクト操作環境の至る所で（ＯＲＢを介して）クライアントがアクセスできる性質もつ一方、Ｃ＋＋オブジェクトやＳｍａｌｌ Ｔａｌｋオブジェクトのような非分散オブジェクトは、共通のオブジェクトとして同一のプロセス中の常駐クライアントのみが利用可能である。当業者は、文脈から、または、明示的な記述からこれら区別できるであろう。

【００３２】クライアントがオブジェクトをコールする場合、オブジェクトに応じたアドレス情報が必要となる。非分散オブジェクト指向操作環境では、クライアントは、同一のアドレス空間内、すなわち、同一のプロセス内にあるオブジェクトとのみ会話することができる。したがって、通常、アドレス情報は、目標オブジェクトが存在するアドレス空間を示すポインタそのものである。

【００３３】しかしながら、分散オブジェクト操作環境においては、クライアントは、求める目標オブジェクト（リモートオブジェクトの場合）に関するオブジェクトレファレンスを持っていなければならない。オブジェクトレファレンスは、本来的に、目標オブジェクトをクライアントに示すために十分なアドレス情報を含んでいる。オブジェクトレファレンスアドレス情報は、直接的なアドレス情報かもしれない。例えば、直接的なアドレス情報は、ホストコンピュータのネットワークアドレス、サーバープロセスのネットワークポート番号、およ

び、目標オブジェクト識別子を含むであろう。しかし、アドレス情報は間接的なものかもしれないし、直接的なアドレス情報と間接的なアドレス情報との組合せかもしれない。

【００３４】アドレス情報については、異なる種類のオブジェクトレファレンスの好適な実施例とともに、本出願人の別出願である、米国特許出願「弁理士書類番号：ＳＵＮ１Ｐ０２５／Ｐ７２１、発明の名称：Methods, Apparatus, and Data Structures for Managing Objects、発明者：Brownell 他」に詳細に記載されている。更に、分散オブジェクト操作環境でのオブジェクト間の接続の形成と終結とに関する好適な実施例が、「弁理士書類番号：ＳＵＮ１Ｐ０１８／Ｐ７１５、発明の名称：Methods and Apparatus for Managing Connections for Communicationamong Objects in a Distributed Object System、発明者：Brownell 他」に記載されている。

【００３５】クライアントが目標オブジェクトをコールする場合、目標オブジェクトがリクエストされたサービスを実行できることを保証するために、特定の手順が実行されなければならない。こうした手順は、目標オブジェクトの識別および配置、（もし、必要ならば）目標オブジェクト側でのサーバーオブジェクトの起動、（もし、必要ならば）目標オブジェクトの活性化、そして最後に、目標オブジェクトとの接続を形成してコールを送る。ＯＲＢは、オブジェクトアダプタ（ＯＡ）とともに、クライアントと目標オブジェクトと共働して、これらのサービスを実行する。

【００３６】本発明の好適な実施例では、透明性のある不変性機構が、オブジェクト開発者が分散オブジェクトを開発するために使用するオブジェクト開発フレームワークによって、自動的に実装される。こうしたフレームワークは、本出願人の米国特許出願「弁理士書類番号：ＳＵＮ１Ｐ０２２／Ｐ７１９、発明の名称：Methodsand Apparatus for Generation and Installation of Distributed Objects ona Distributed Object System、発明者：Snyder 他」に詳細に記載されている。ひとつの好適なオブジェクト開発フレームワークがＳｕｎＳｏｆｔ（Ｍｏｕｎｔａｉｎ Ｖｉｅｗ，ＣＡ）から提供されており、利用可能である。継承のオブジェクト指向原理（obuject oriented principles）が、分散オブジェクトに対して透明性を持って本発明を提供するためのオブジェクト開発フレームワークを可能とするために使用されるであろうことは、当業者には明らかであろう。

【００３７】オブジェクト「クラス」は、オブジェクトがそれから作成され得るテンプレートである。それは、クラス内の全てのオブジェクに共通の振る舞い（behavior）や属性（attributes）を特定するために使用される。現存しているクラスから新たなクラスを定義する機構は、「インヘリタンス（inheritance；継承）」である。クラスの「サブクラス」は、親の動作を継承する。よく知られているように、「インヘリタンス」は、再使用可能性を容易にする機構である。

【００３８】図２は、本発明の一実施例に従って、オブジェクトの作成の一つの可能なフローを示している。データスキーマ（data schema）１７０は、オブジェクト開発者によって記述される。データスキーマ１７０は、全てのオブジェクト変数の構造と性質とを定義し、記述する。そして、データスキーマ１７０は、データオブジェクトクラス１７４を作成するために、データコンパイラ１７２によって処理される。したがって、データオブジェクトクラス１７４は、開発されつつあるオブジェクトに関するデータの振る舞いと属性を記述している。特に、データオブジェクトクラスはデータオブジェクトポインタの値を含んでおり、このことの重要性を以下に詳述する。

【００３９】実装（implementation）１７６は、開発されつつあるオブジェクトの振る舞いと属性とを定義するオブジェクト開発者によって記述される。実装１７６は、コンパイル済実装１８０を作成するために、実装コンパイラ１７８によってコンパイルされる。データスキーマ１７０、データオブジェクトクラス１７４、実装１７６、コンパイル済実装１８０は、通常は、夫々、オブジェクト開発者またはコンパイラによって書かれたファイルであることに注意されたい。

【００４０】特に、コンパイル中に、データオブジェクトを参照する記載がデータオブジェクトに対するポインタによって置き換えられることは、当業者には理解されるであろう。例えば、ｘがサーバントであり、ａがデータオブジェクトメンバである、「ｘ．ａ」という記載は、「ｄｏｐ」がデータオブジェクトクラス内でその値が見出されるデータオブジェクトポインタである、「ｘ．ｄｏｐ→ａ」という記載に、コンパイル中に置き換えられる。したがって、サーバントがデータオブジェクトを継承するならば、データオブジェクトポインタは、適当なデータオブジェクトに対してデータ操作（すなわち、読み出し操作および書き込み操作）を間接的に指示する。

【００４１】データオブジェクトクラス１７４とコンパイルされた実装１８０は、コンパイラの他の段階で処理され、オブジェクトコンストラクタ１８４を作成する。当業者には理解されるように、オブジェクトコンストラクタ１８４を実行すると、所望のオブジェクトのインスタンスが作成される。

【００４２】本発明の一実施例のサーバントオブジェクト１５０について、図３を参照して説明する。図３に示すように、分散オブジェクトクラスのサーバントオブジェクト１５０は、一時データ１５２、データオブジェクトポインタ１５４、およびデータオブジェクト１５８を含む。一実施例では、サーバントはデータオブジェクト

を継承する。したがって、当業者には明らかなように、データオブジェクトポインタ１５４の値はデータオブジェクトクラス内に保持される。これにより、もちろん、全ての作成されたサーバントが同一のデータオブジェクトクラスからデータオブジェクトポインタの値を継承するので、労力を軽減して多数のサーバントの作成をすることができる。

【００４３】サーバントオブジェクト１５０は、本発明の一実施例に関する分散オブジェクトの１つのインスタンスである。オブジェクト指向プログラミングの当業者には明らかなように、クラスの新たなインスタンスが作成されると、インスタンス変数に関するメモリが割り当てられる。図３には、サーバントオブジェクト１５０のメモリ割り当てが示されている。一時データ１５２は、オブジェクトが一時データとしてのみ使用することが必要な状態（例えば、変数）を含むサーバントオブジェクト１５０のメモリ部分である。また、一時データは、サーバントオブジェクト１５０のこのインスタンスによってのみ使用されるスクラッチパッドとして想定される。データオブジェクト１５８は、オブジェクト開発者が不変状態として使用を意図しているインスタンスのメモリの部分である。メモリのこの領域は、上記のデータオブジェクトクラスから継承されたデータオブジェクトポインタによって持しされる。

【００４４】まず、サーバントオブジェクト１５０が作成される場合、オブジェクトコンストラクタ１８４のようなオブジェクトコンストラクタが、初期化状態（データオブジェクトが割り当てられたメモリ中がランダムな値のまま放置されている、または、０に初期化されている、等）中の一時データ１５２およびデータオブジェクト１５８とともにサーバントオブジェクト１５０を作成する。（作成期間に実行される）初期化機能および（活性化期間に実行される）活性化機能の双方は、要請された初期化を実行しなければならない。上述のように、このことは、プログラマに不変性を実装するために、必要な機能を提供する負荷を担うことを要求する。

【００４５】しかし、本発明の好適な実施例によれば、（図１のように）データオブジェクト１５８を示す、元のデータオブジェクトポインタ１５４は、作成中あるいは活性化中に、不変データオブジェクトをコールしたデータオブジェクト１５８のレプリカを保有するメモリの領域を指示する不変ストレージマネージャデータオブジェクトポインタ（a persistent storage manager data object pointer）１５４に置き換えられる。オブジェクトが作成されるか、活性化されたとき、レプリカ不変データオブジェクト１６２は、不変データベース１６４のような不変ストレージの外部に写像される。不変データオブジェクト１６２の格納形式は、作成期間に生成され、オブジェクトが削除されまれで、不変データベース内で維持される。以下、図７から図１０を参照して、本発明に従う透明性のある不変性を実装するオブジェクトの作成、活性化、および削除に関する好適な方法を説明する。

【００４６】好適な実施例では、不変データストレージ１６４は、不変ストレージマネージャによって管理される。不変ストレージマネージャの好適な実施例は、米国出願「弁理士書類番号：ＳＵＮ１Ｐ０３６、発明の名称：Methods and Apparatus for Managing a Database in a Distributed Object Operating Enviroment、発明者：Hapner　他」に記載されている。

【００４７】本発明で考察されているような分散オブジェクトは、（ＯＲＢおよび／またはホストコンピュータによって）コンピュータプロセスのもとで実行される。コンピュータプロセスは、良く知られた共通のフレームワークを提供し、その下で、コンピュータシステムは、複数の異なるスレッドを実行する。コンピュータプロセスは、コンピュータシステムの分割された領域として考えることができる。

【００４８】実際に、プロセスは、通常、アドレススペース（すなわち、当該プロセスのみが割り当てられたメモリ部分）、ファイル記述子の集合、プロセス識別番号、および１つ以上の実行のスレッドを含む。本発明の一実施例で考察されているような多重スレッドシステムでは、複数のスレッドを単一のプロセスないで同時に実行することができる。スレッド、多重スレッドシステム、および同時性の詳細に関しては、ＳｕｎＳｏｆｔ社発行の「Dr. Robert Hagmann 著、" Concurrency Within DOE Object Implementations"、Version 0.91, May 27, 1993」を参照されたい。

【００４９】図４に、本発明の一実施例に応じた、多重スレッドプロセス１００を含む本発明の実施例を示す。プロセス１００はコンピュータ３０のようなコンピュータ上に実装され、スレッド１０２のような多重スレッド、不変および一時的メモリを有する割り当てられたメモリ１０４、ファイル識別子１０６、および、オブジェクト１０８のような少なくとも１つのオブジェクトを含む。オブジェクト１０８は、状態１１０とコード１１２とを含む。オブジェクト１０８は、通常、状態１１０およびコード１１２によって定義され。コード１１２は、本質的には、オブジェクトが実行する動作命令である。したがって、状態１１０は、実行可能ではないコードである、残りの部分である。明らかなように、状態１１０は、一般的に、本発明の透明性のある不変性が目指しているものである。本発明のいくつかの実施例では、不変ストレージマネージャが、複数のプロセス中に存在する複数の分散オブジェクトにとってアクセス可能な単一のデータベースを保守する。

【００５０】本発明の好ましい実施形態において、分散オブジェクト、コンピュータプロセスおよび分散オブジェクトのクライアントは、ネットワークによって相互に

連結された1台以上のコンピュータに配置される。ネットワークは、いくつかの好ましい形態を取り得る。例えば、図5に、代表的なネットワーク編成10を示す。ネットワーク編成10はネットワーク14に結合されたコンピュータ12を備える。ネットワーク編成10は、別のコンピュータ18，20，および22に加えてサーバ、ルータ等16を備え、データや命令がネットワーク化されたコンピュータ間で転送できる。こうしたコンピュータネットワークの設計、構成、および実装は当業者には周知である。

【0051】図5のコンピュータ12，16，18，20，および／または22の構成を、図6に示す。コンピュータ30は、ランダムアクセスメモリ（RAM）34に双方向的に、リードオンリメモリ（ROM）36に単方向的に接続された中央処理装置（CPU）32を備える。通常、RAM34は、CPU32上で現在作動中のプロセスに関する分散したオブジェクトおよびそれらのオブジェクトに関連するデータと命令を含む、プログラミング命令およびデータを格納している。ROM36は、通常、コンピュータが機能を実行するために用いる基本的動作命令、データ、およびオブジェクトを格納ている。更に、ハードディスク、CD ROM、磁気光学（フロプティカル）ドライブ、テープドライブ等の大容量記憶装置38が双方向的にCPU32に接続されている。一般には、大容量記憶装置38は、アドレス空間がCPU32によって、例えばバーチャルメモリ等としてアクセス可能であるにもかかわらず、CPU32によって、通常は頻繁に用いられない追加のプログラミング命令、データ、およびオブジェクトを格納する。上記の各コンピュータは、通常、更に、キーボード、ポインタ装置（すなわち、マウスやスタイラス）等の入力媒体を含む入出力源210を備える。また、ネットワーク接続を介して、追加の大容量記憶装置（図示せず）をCPU202に接続することができる。上記ハードウェア要素、ソフトウェア要素、およびネットワーキング装置は、当業者にとって周知の標準設計構成である。

【0052】図7を参照して、本発明における透明性のある不変性を有するオブジェクトを作成する方法を説明する。透明性のある不変性の方法200は、種々のシステムで使用される。例えば、この方法は、CORBA仕様に基づく分散オブジェクト操作環境における使用に適している。方法200は、オブジェクトクリエータがオブジェクトの作成のためにコールされたとき、ステップ201で開始する。オブジェクトの作成のために、オブジェクトのファクトリクラスが利用されることは、当業者には理解されるであろう。通常、オブジェクトインストールにあたって、ファクトリクラスの単一のインスタンスが作成され、適当なクライアントによってアクセス可能なサーバプロセス内に置かれる。いくつかの場合、サービスをリクエストしようとするクライアントは、使用するオブジェクトのインスタンスを作成するため、このファクトリサーバントオブジェクト（factory servantobject）をコールするかもしれない。他の場合、ORBが、サービスをリクエストしているクライアントに応答して、クリエータをコールするかもしれない。

【0053】ステップ202で、サーバントオブジェクト150が作成される。通常、この動作は、上述のように、オブジェクトコンストラクタの実行によって終了する。ステップ202において、メモリの連続部分が、一時データ152、データオブジェクトポインタ154、およびデータオブジェクト158に関するメモリを含むサーバントオブジェクト150に割り当てられる。上述のように、データオブジェクトポインタ154に関する値は、データオブジェクトクラスから継承される。

【0054】割り当てられた当初、データオブジェクトポインタ154はデータオブジェクト158を指している。そして、ステップ204で、不変データオブジェクト162が作成される。好適な実施例では、（オブジェクト開発者によってプログラミンされたデータオブジェクトのレプリカである）不変データオブジェクト162は、不変ストレージマネージャによって保守される不変データベース164内のストレージから写像される。ステップ204においては、不変データオブジェクト162は、RAM34のような一時メモリ内に写像される。不変データオブジェクト164をRAM34内へのコピーは、不変ストレージ内で保守されるよりも、速い読み出し／書き込み操作を可能とすることが、当業者には理解されるであろう。

【0055】次に、ステップ208で、オブジェクトアダプタオブジェクト（object adapter object）が獲得される。このステップは、ORBにサーバントオブジェクト150を登録すること、および、リターン時にオブジェクトレファレンスを受けることを含む。オブジェクトアダプタオブジェクトの生成は、周知の方法で実行される。例えば、CORBA標準は、ステップ208で要求されるような分散オブジェクトの生成に関するガイドラインを提供している。次いで、ステップ210で、オブジェクトレファレンスが、サーバントオブジェクト150のホストコンピュータ上のORBによって保守されるオブジェクトレファレンスデータ内に格納される。ステップ212で、初期のデータオブジェクトポインタ154は、先のステップ204で生成された不変データオブジェクト162を指示する不変データポインタ154に置き換えられる。ステップ212の一実施例を、図8を参照して、後に詳細に説明する。

【0056】次に、ステップ214で、初期化手順が実行される。これは、変数の初期値の設定のステップ、および、不変データ内での読み出しステップを含んでいる。初期化手順（少なくとも一部）は、オブジェクト開発者によって記述され、各オブジェクト作成に対してユ

ニークとなるであろう。したがって、活性化期間よりも、作成初期化ステップ２１４の期間のほうが、異なる機能が実行されるであろう。ここで、オブジェクトが、作成され、活性化され、そして、サービスに関するリクエストを受けることができるようになり、ステップ２１６で、オブジェクトレファレンスが、作成方法２００を開始したクライアントに返される。

【００５７】図８を参照して、図７のデータポインタ置換ステップ２１２の一実施例を詳細に説明する。図８に示す実施例は、Ｃ＋＋やＳｍａｌｌ　Ｔａｌｋのような、オブジェクト指向プログラミング言語で使用するのに好適である。こうした言語は、元々、分散オブジェクト操作環境や透明性のある不変性のために設計されていないので、図８の方法は、現在使用可能なプログラミングフレームワークにおいて、本発明の方法を行うことを可能とする、１つの手法を提供する。他のプログラミング言語が使用される場合には、データポインタ置換ステップ２１２のメカニズムを適当に変形すればよいことは、当業者には理解されるであろう。

【００５８】図８のデータポインタ置換ステップ２１２は、データポインタ１５４の値が識別されたとき開始する。図２を参照して、上記で説明したように、分散オブジェクトの特定の特徴は、「親（parent）」クラスから「継承（inherited）」される。この場合では、データポインタ１５４が、データオブジェクトクラスから継承される。現状のオブジェクト指向プログラミング言語（例えば、Ｃ＋＋やＳｍａｌｌ　Ｔａｌｋ）の構造および継承性（inheritance）の性質の結果として、データポインタ１５４のアドレスは「隠されている（hidden）」か、または、分散オブジェクトにとって利用不可能である。更に、データオブジェクトポインタ１５４は、分散オブジェクトのライフサイクルで一定ではない。この一因は、異なる活性化／非活性化サイクル期間中、夫々の活性化フェーズで、分散オブジェクトが、異なるメモリ空間（すなわち、異なるアドレス）に存在することである。一般に、活性化時にオブジェクトに割り当てられたメモリ空間は、ＯＲＢによって知られているが、ＯＲＢは、データポインタ１５４の特定位置を知らないことにある。

【００５９】しかし、周知のように、適切な「拡張（widening）」機能を実行することにより、データポインタ１５４の値をデータオブジェクトクラスから獲得することができる。ステップ２５２で、データポインタ１５４の値が獲得されると、ステップ２５４で、ステップ２５２で見出した値と同一の値を含むアドレス空間を見出すために、分散オブジェクトのメモリ位置を探す。ステップ２５４で見出されたアドレス情報は、データポインタフィールド１５４内の値を新たなデータオブジェクトポインタ１５４に置き換えるステップ２５６を可能とする。一実施例において、データオブジェクトポインタ１５４は、不変メモリ中のデータを保守する不変ストレージマネージャによって提供される。不変ストレージマネージャのより詳細に関しては、米国出願「弁理士書類番号：ＳＵＮ１Ｐ０３６／Ｄ４２２」を参照されたい。データオブジェクトポインタ１５４が置換されると、図８の方法が終了する。

【００６０】探査ステップ２５４の実行中において、値判定ステップ２５２で得られた値が、分散オブジェクトに割り当てられたメモリ空間内でユニークではないというリスクが、僅小ではあるが存在することに指摘しなければならない。したがって、いくつかの実施例においては、唯一性を保証するために、メモリ空間の完全な探査はステップ２５４で実行されるであろう。しかし、上記のリスクは小さいので、殆どの実施例においては、付加的な保証ステップは不必要と信じられる。理解されるように、図８の方法の自明な変形が、異なるオブジェクト指向プログラミング言語を使用する場合に適当である。例えば、いくつかのインスタンスでは、オブジェクト指向プログラミング言語が、直接的に、データポインタ１５４のアドレスを獲得する機構を提供しているかもしれない。こうした場合には、ステップ２５４は不要であり、上記のリスクは除去される。

【００６１】図９を参照して、本発明での、オブジェクトを活性化する方法３００を説明する。ステップ３０２で、オブジェクトに関する発動（invocation）が受信される。発動は、オブジェクトが存在するサーバプロセスによって受信されるであろう。次いで、ステップ３０４で、サーバプロセスが、オブジェクトが活性であるか否かが判定される。既に、オブジェクトが活性であれば、不変データオブジェクト１６２は既に配置されており、データオブジェクトポインタは、上記のように既に置換されている。したがって、本発明に関連する活性化のステップは既に実行されており、活性化方法３００は終了する。

【００６２】オブジェクトが非活性であれば、ステップ３０６で、コンストラクタがサーバントオブジェクト１５０の作成をコールする。ステップ３０６においては、一時データ１５２、データオブジェクトポインタ１５４、およびデータオブジェクト１５８を有するサーバントオブジェクト１５０にメモリの連続部分が割り当てられる。

【００６３】割り当てられた当初、データオブジェクトポインタ１５４はデータオブジェクト１５８を指している。図９のステップ３０６が、図７のステップ２０２に類似していることに注目されたい。そして、ステップ３０８で、不変データオブジェクト１６２が、不変データベース１６４内のストレージから検索され、ＲＡＭ３４のような一時メモリ内に写像される。不変データオブジェクト１６４をＲＡＭ３４内へのコピーは、不変ストレージ内で保守されるよりも、速い読み出し／書き込み操

作を可能とすることが、当業者には理解されるであろう。このステップは、図7のデータオブジェクト作成ステップ２０４と同様であることに注目されたい。

【００６４】しかし、検索ステップ３０８においては、データオブジェクトのレプリカが既に作成されており、それを不変データベース１６４から直接読み出すことができる。データオブジェクト１６２が、アドレスが知られたメモリ内に写像されたとき、すなわちステップ３１０で、元のデータオブジェクトポインタ１５４が、上述の方法を使用して、不変データオブジェクト１６２を指すデータオブジェクトポインタに置き換えられる。ポインタ置換ステップ３１０の後、ステップ３１２で、活性化の初期化ステップが実行される。いくつかの実施例では、活性化フックが、オブジェクト開発者によってプログラムされ、活性化にあたって実行さなければならない全ての必須機能を定義している。活性化フックは、図7のオブジェクト作成２００の初期化ステップ２１４で実行される初期化については不要であることに注意されたい。

【００６５】図１０を参照して、本発明の一実施例に関する、オブジェクトを削除する方法３５０について説明する。最初のステップ３５２で、クライアントが、オブジェクトに対して、「廃棄（destroy）」操作を発動するコールを行う。これは、（たとえ、外部で開始されも）自己実行動作であるオブジェクト削除を有するプリグラミング手法に聞こえるで有ろう。これは、オブジェクト開発者によって定義された、秩序立ったシャットダウンと削除を保証する。したがって、オブジェクトは、インターフェイスを介してクライアントが利用可能な「廃棄」のような操作を有する（オブジェクトを削除するオブジェクトの完全独立性とは異なっている）。オブジェクトは、非活性や置換のような様々な理由で削除される。オブジェクト（または、オブジェクトが置かれたプロセス）が非活性となる場合に、対称的にオブジェクトを削除する好適な方法は、米国特許出願「弁理士書類番号：ＳＵＮ１Ｐ０３５／Ｐ７６９、発明者：Snyder他」に開示されている。

【００６６】オブジェクトが自己廃棄（self-destroy）の要求をうけると、ステップ３５４で、オブジェクトは、内部削除操作を実行する。これは、新たなリクエストをブロックしつつ、係属中のサービスリクエストを完了させ、削除が完了した旨を表示する最終メッセージを通知する、系統的な非活性化ステップを含む。オブジェクトを非活性化する好適な方法の一つが、米国特許出願「弁理士書類番号：ＳＵＮ１Ｐ０２６／Ｐ７１８、発明の名称：Methods and Apparatus for Managing Deactivation and Shutdown of a Server、発明者：Yeturu Aahlad」に開示されている。次に、ステップ３５６で、オブジェクトに関連する不変データが削除される。このステップは、廃棄操作中のオブジェクト自身によって実行されるかもしれないし、オブジェクトが廃棄操作を実行した後に不変ストレージマネージャによって行われるかもしれないし、オペレーティングシステムのような他のコンピュータエンティティによって実行されるかもしれない。オブジェクトおよびその不変データが削除されると、図１０の方法は終了する。

【００６７】以上、本発明のいくつかの実施例を説明したが、本発明はその精神と範囲から逸脱することなく他の多くの具体的な形態において実施できるものと解釈すべきである。例えば、図の２の実施例は、単に、オブジェクト開発手順の１例を提供するにすぎない。しかし、多くの他のオブジェクト開発手順が、本発明の方法および装置を実装のために使用できることが、当業者には理解されるであろう。

【００６８】図９に示した実施例において、透明性のある不変性の方法３００は、クライアントがオブジェクトをコールすることにより開始される。しかし、他の適当な事象が図９の方法を開始するかもしれない。例えば、分散オブジェクトは自己活性化するかもしれない。更に、データポインタ置換ステップ３１０の実装の前に、他の判定ステップが出現するかもしれない。例えば、以前の活性化のデータオブジェクトを無視しなければならない活性化環境かもしれない。この場合、異なるポインタが分散オブジェクトに返され、不変データを格納するための「新たな」位置とともにポインタが供給されるかもしれない。したがって、提示された例は説明のためであって、制限するためではなく、本発明はここに示された詳細事項に限られるのではない。

【図面の簡単な説明】

【図１】従来技術のサーバントオブジェクトのインスタンス変数のメモリ割り当ての説明図である。

【図２】オブジェクトコンストラクタのコンパイルに関するパラダイムの説明図である。

【図３】本発明のサーバントオブジェクトのインスタンス変数のメモリ割り当ての説明図である。

【図４】プロセスの説明図である。

【図５】相互接続された種々のコンピュータを備えるコンピュータネットワークシステムの構成図である。

【図６】図５におけるコンピュータの構成図である。

【図７】分散オブジェクトのインスタンスを作成する方法のフローチャートである。

【図８】図７のステップ２１２の詳細フローチャートである。

【図９】オブジェクトを活性化する方法のフローチャートである。

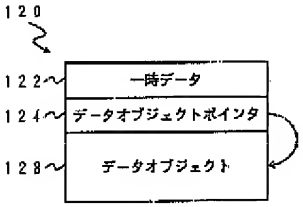
【図１０】オブジェクトを削除する方法のフローチャートである。

【符号の説明】

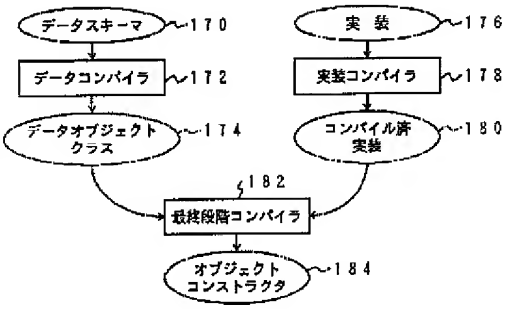
１２，１６，１８，２０，２２…コンピュータ、１４…ネットワーク接続、３０…コンピュータ、３２…中央処

理装置（CPU）、34…RAM、36…ROM、38…大容量記憶装置、40…入出力源、100…コンピュータプロセス、102…スレッド、104…メモリ、106…ファイル、108…オブジェクト、110…状態、112…コード、150…サーバントオブジェクト、152…一時データ、154…データオブジェクトポインタ、158…データオブジェクト、162…不変データオブジェクト、164…不変データベース。

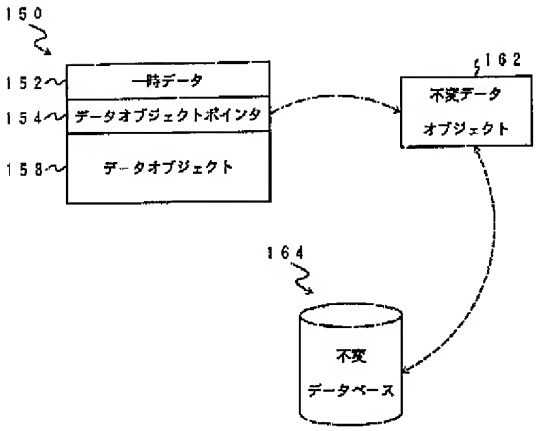
【図1】



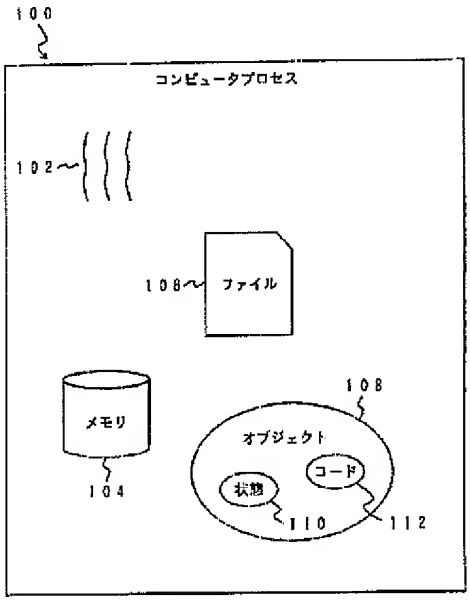
【図2】



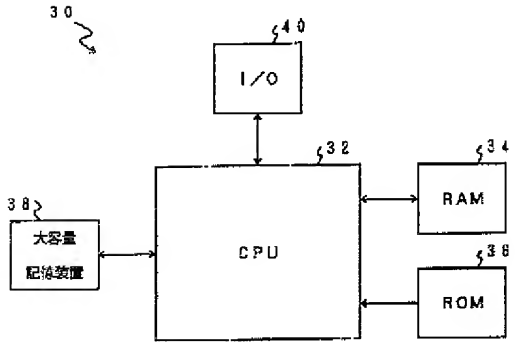
【図3】



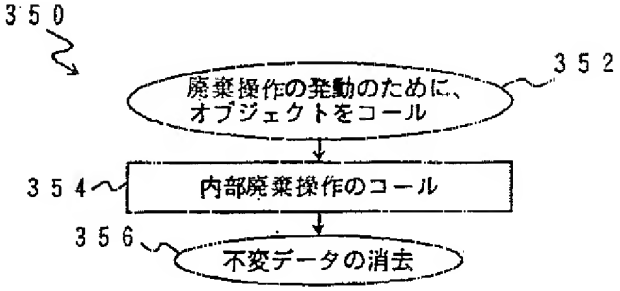
【図4】



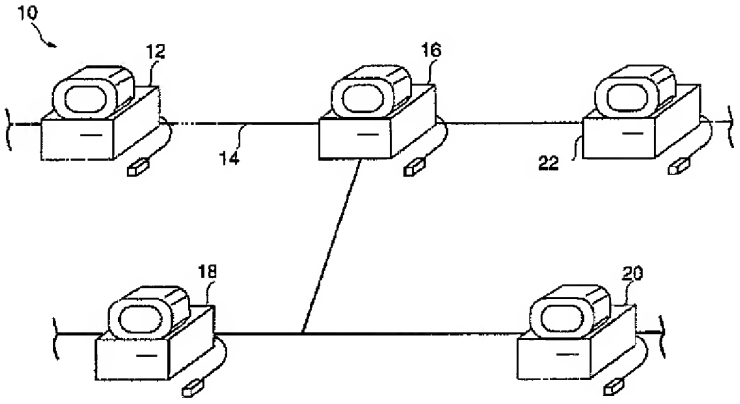
【図6】



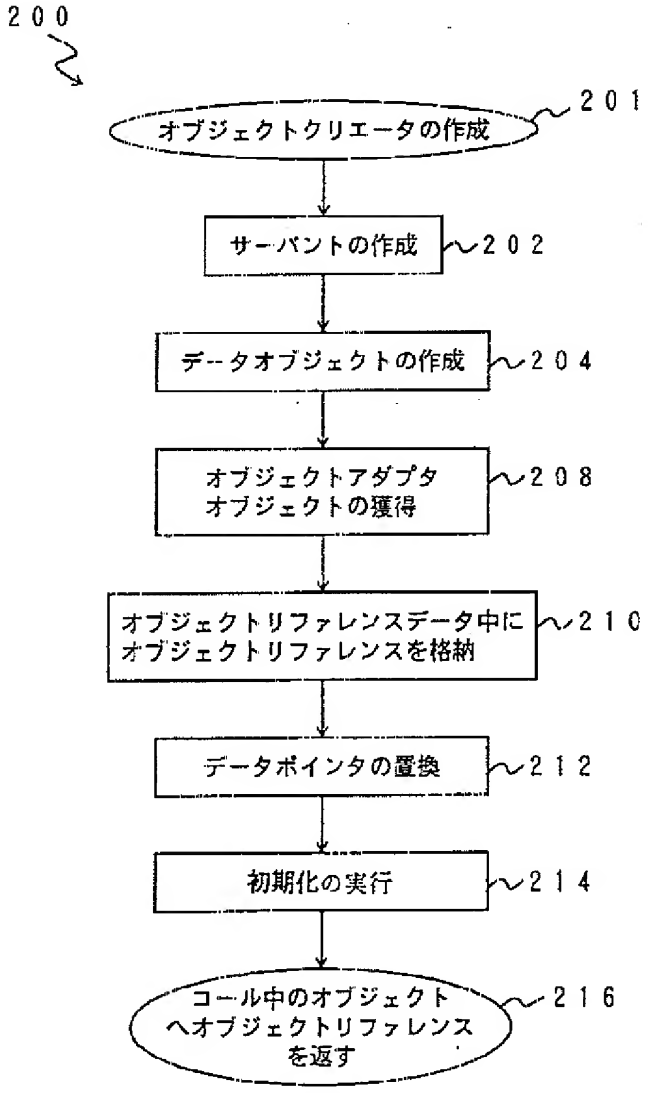
【図10】



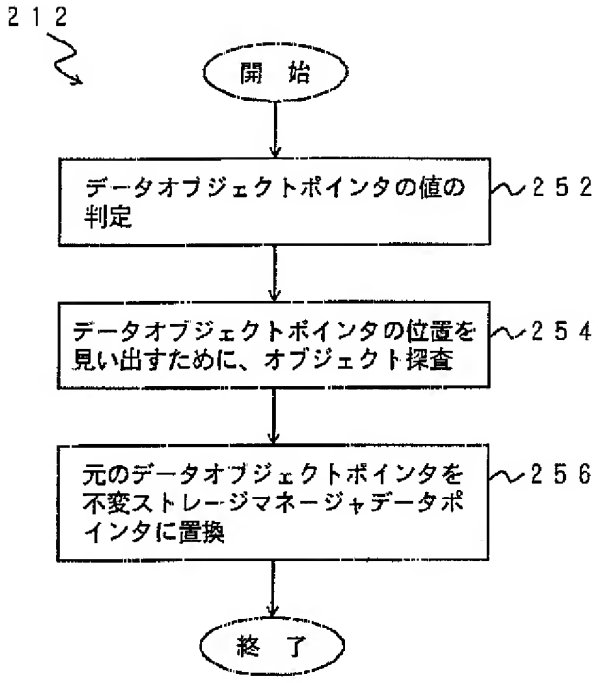
【図5】


10
12
16
14
22
18
20

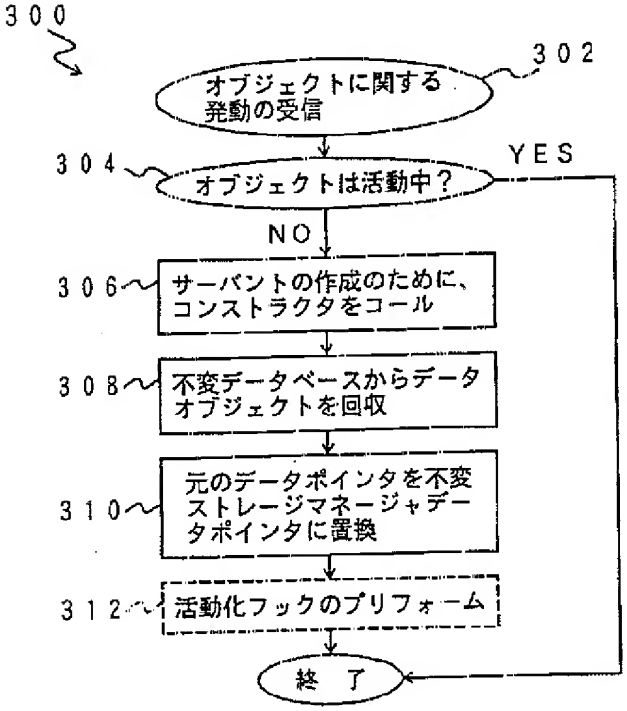
【図7】


200
オブジェクトクリエータの作成
201
サーバントの作成
202
データオブジェクトの作成
204
オブジェクトアダプタ
オブジェクトの獲得
208
オブジェクトリファレンスデータ中に
オブジェクトリファレンスを格納
210
データポインタの置換
212
初期化の実行
214
コール中のオブジェクト
へオブジェクトリファレンス
を返す
216

【図8】


212
開 始
データオブジェクトポインタの値の判定
252
データオブジェクトポインタの位置を見い出すために、オブジェクト探査
254
元のデータオブジェクトポインタを不変ストレージマネージャデータポインタに置換
256
終 了

【図9】


300
302
オブジェクトに関する発動の受信
304
オブジェクトは活動中？
YES
NO
306
サーバントの作成のために、コンストラクタをコール
308
不変データベースからデータオブジェクトを回収
310
元のデータポインタを不変ストレージマネージャデータポインタに置換
312
活動化フックのプリフォーム
終 了

フロントページの続き

(51) Int Cl.[6]	識別記号	庁内整理番号	FI		技術表示箇所
G06F 15/16	370		G06F 15/16	370N	

(72)発明者　アラン　スナイダー
アメリカ合衆国，　カリフォルニア州
94306，　パロ　アルト，　ブライアウッド　ウェイ　4160